AN INTRODUCTION TO PRISM SIGNAL PROCESSING APPLIED TO SENSOR VALIDATION

Manus P.Henry

University of Oxford

manus.henry@eng.ox.ac.uk

Abstract

The Internet of Things (IoT) [1] and Industrie 4.0 [2] propose substantial increases in the deployment of sensors into a diverse range of environments. The challenges are considerable: local computational power must be efficiently and flexibly deployed both to perform current metrological tasks and to reconfigure/redesign the signal processing flow as monitoring requirements evolve over time.

This paper introduces the Prism (precise, repeat integral, signal monitor), a new type of signal processing block, as a contribution towards to the challenges of 21st Century metrology. The Prism acts as a fully recursive, dual output, FIR filter: the computational burden is low and independent of data window length. Prism design is trivial, so that networks of Prisms can be assembled, whether at design time or autonomously in real time, to carry out a wide range of metrological tasks. Prism-based trackers for the frequency, phase and/or amplitude of a sinusoid perform close to the Cramer-Rao Lower Bound (CRLB) for SNRs down to 0 dB.

Two aspects of Prism signal processing: low computational and negligible design costs, provide useful tools for the development of simple sensor validation techniques [3]. A simulation example demonstrates how Prism signal processing can be used to autonomously detect, track, and compensate for an undesired frequency component in a frequency-based sensor.

1. The Prism

The Prism (Fig. 1) is a signal processing block accepting an input time series s(t) and generating one, or more usually two, output time series, $G_s(t)$ and/or $G_c(t)$. Internally, it is structured as two layers of integration blocks, where each input signal is multiplied by a modulating sine or cosine function of characteristic frequency m and harmonic number h (a small integer, often 1), and the resulting product is integrated over the last $1/m$ seconds. The two outputs result from a sum and a difference of second stage integrals. The Prism can be viewed as a pair of FIR filters operating over a window of the input data $s(t)$ of total duration $2/m$. Unusually, the Prism calculations can be performed recursively, so that the computational effort needed per sample is small, and is independent of the Prism window length. This facilitates high data throughout for a given computational budget. Another significant advantage of the Prism is that the filter 'coefficients' are simply the linearly spaced sine and cosine values of the modulation functions. Accordingly, the computation requirement to 'design' a Prism with desired m and h values is very low, and so it is possible to instantiate new Prism-based signal processing schemes in real time on resource-limit devices using simple rules. Further details of the Prism will be given in later publications.

For a steady sinusoidal input with amplitude A, frequency f and initial phase ϕ_i (i.e. the phase at t = 0)

$$s(t) = A\sin(2\pi ft + \phi_i)\,, \qquad (1)$$

the Prism outputs $G_s(t)$ and $G_c(t)$ are given by:

$$G_s(t) = A\,\mathrm{sinc}^2(r)\frac{r^2}{r^2-h^2}\sin(\phi(t)-2\pi r) \qquad (2)$$

and

$$G_c(t) = A\,\mathrm{sinc}^2(r)\frac{hr}{r^2-h^2}\cos(\phi(t)-2\pi r)\,, \qquad (3)$$

where the true instantaneous phase $\phi(t) = 2\pi ft + \phi_i$; $r = f/m$, the frequency ratio; and sinc(x) is the normalized sinc function. The Prism outputs have a linear phase delay $2\pi r$, while the gains of $G_s(t)$ and $G_c(t)$ for $h = 1$, labelled Γs and Γc respectively, are shown in Fig. 2; these exhibit a generally low pass characteristic with periodic notches at multiples of m, including at zero hertz.

The two Prism outputs $G_s(t)$ and $G_c(t)$ are orthogonal (i.e. a sine/cosine pair), and other than a scaling factor h/r, they form an analytic function from which sample-by-sample estimates of frequency, amplitude and phase may be derived. In other words, given an input signal with unknown sinusoidal properties, which is passed through a Prism with characteristic frequency m, the sinusoidal parameters may be estimated based on values of $G_s(t)$ and $G_c(t)$. One means of tracking a single sinusoid is the Recursive Signal Tracker (RST) block which employs a single Prism and recent history of $G_s(t)$ and $G_c(t)$.

Reference [4], co-authored with Russian colleagues at SUSU in Chelyabinsk, outlines the RST calculation, along with its application to a pressure sensor validation problem. The mechanical integrity of the pressure sensor is tested by regular excitation from ultrasonic pulses. The response of the pressure sensor structure to each pulse takes the form of multiple exponentially decaying sinusoids combined into a single signal, where the properties (frequency, initial amplitude, decay rate) of each sinusoid are used to detect faults such as the fouling of the sensing membrane. A variety of Prism-based techniques are used to isolate individual frequency components for tracking. These include low pass and bandpass filtering, conventional static notch filtering, and a new technique called dynamic notch filtering, whereby one or

more frequency components can be removed from a signal and where the notched frequencies can be selected in real time. This entails forming linear combinations of Prism outputs [4], where the Prisms share a common characteristic frequency m, but use different harmonic values h. Linear combinations of the same Prism outputs, but using different weightings, have the effect of notching out different frequency components, and so it is possible to 'split' a multi-component signal into a set of individual signals, each carrying only a single frequency component, which may then be tracked using an RST. As illustrated in [4], such signal splitting may be implemented sample-by-sample with low computational cost.

The Prism thus provides a useful toolkit for basic signal processing tasks where the compute and design requirements are low, and hence is suited to the challenges of metrology in the 21st Century, where the Internet of Things provides ubiquitous, adaptable sensing.

2. Sensor Validation Example

An example is now given of how Prism-based techniques may be used to carry out the following tasks: tracking a signal consisting of a single frequency component; detecting the presence of an anomalous frequency component in the signal; and then instantiating additional Prism-based signal processing to isolate and track both the original frequency component and the anomalous frequency component. The resource requirements for both the instantiation (including the design) and the operation of this validation scheme are sufficiently low that it is suited for the low cost, autonomous sensors of the IoT.

Figure 3 shows an initial arrangement where an input signal, for example from a resonant transducer, is being tracked by a RST, which generates sample-by-sample estimates of frequency, amplitude and/or phase, under the assumption that the input signal has only a single frequency component to be tracked. These parameters may in turn be mapped onto appropriate engineering units to

calculate the measurand under consideration (for example, pressure, temperature, flow rate etc). For the purposes of this simulation, the underlying sensing technology is not important. Rather, this is a generic and simplified example that may be mapped onto a number of specific measurement technologies.

In the simulation study that follows, the sample rate fs = 51.2 kHz, and the RST uses m = 200 Hz for tracking the input signal. For the first 100 seconds of the simulation, only a single frequency component, denoted the Primary component, is present. This has a frequency of 81 Hz and an amplitude of 0.5 V. The input signal includes white noise with a standard deviation of 1e-5 V. After 100 seconds, an additional frequency component, denoted the Interference component, is added to the input signal, thus simulating the onset of an unexpected anomaly, as might be caused by an internal fault or some external interference. In this example the additional frequency component has a constant frequency of 53 Hz and an amplitude of 1 mV (1e-3 V). Note that the signal processing design is general and is not tuned to any of the specific parameter values stated here.

Figures 4 and 5 demonstrate the impact such an anomaly has on the RST signal tracking. Figure 4 (upper) shows a time sequence of the input signal, after the onset of the fault. Given the low amplitude of the Interference component there is little observable change in the time series. However in the lower plot the corresponding power spectrum (for the time t = 100 s … 200 s) shows the presence of the Interference frequency component at 53 Hz as well as the Primary frequency components at 81 Hz.

Figure 5 shows the estimate of frequency generated by the RST during the onset of the Interference component. The upper plot shows the time sequence of the frequency estimate: a clear change in behaviour takes place at t = 100 s, when the Interference component is included in the input signal $s(t)$. The lower plot shows the magnitude of the frequency error on a logarithmic scale. Prior to

the onset of the Interference component the frequency error was approximately within the range ± 1e-4 Hz, but after the onset of the Interference component the frequency error range increased to approximately ± 5e-2 Hz. Similar changes are observed in the errors for the amplitude and phase tracking (graphs not shown). Before the 'fault' the amplitude error was within the range ± 5e-6 V, but afterwards the amplitude error range increased to ± 1e-3 V. Similarly, prior to the fault the phase error was approximately ± 1e-5 radians, but afterwards the phase error range increased to ± 1e-2 radians.

Figure 6 demonstrates that the impact of the Interference component on the calculated values of frequency is to introduce a cyclical error. The upper plot shows a close up of the RST frequency time series after the onset of the fault, which shows a repeated cycling pattern. The lower plot shows the power spectrum of the entire RST frequency estimate for t = 100 s to t = 200 s. It is important to appreciate that this is not the power spectrum of the original signal *s(t)*. Rather, the frequency output of the RST is treated as a separate signal in its own right, and the power spectrum is performed upon the resulting time series. Accordingly, the RST frequency output has a large DC (zero hertz) component, corresponding to the true parameter value of 81 Hz, but in addition there are two peaks at approximately 28 Hz and 134 Hz. These are the so-called beat frequencies, found by forming the difference and sum respectively of the two original frequencies (81 Hz – 51 Hz = 28 Hz, and 81 Hz + 53 Hz = 134 Hz). Similar beating behaviour is observed on the amplitude estimate; these symptoms both define the measurement interference problem, and suggest a means of detecting and ultimately compensating for these effects.

Figure 7 shows an extension to the original RST tracker scheme of Figure 3. The RST on the left is comparable to that of Figure 3 but additional signal processing is provided to detect any interference component in the input signal *s(t)*. This includes a second RST which accepts the frequency estimate of the first RST as an input and estimates the frequency of variation of this parameter –

effectively, it is being used to track the (lower) beat frequency. This estimated 'raw' beat frequency is fed into a further signal processing block which calculates a sliding window estimate of its mean and standard deviation. A low standard deviation is likely to indicate the presence of a steady, as opposed to a randomly varying, beat frequency, and hence the presence of an interference component in the signal. The mean beat frequency is used rather than the raw value for later correction calculations as it has reduced noise. A threshold test is applied to the estimated standard deviation, where a low value indicates the presence of an interference frequency. Note that additional testing might include the 'amplitude' of the frequency modulation to ensure that the detection is not triggered by minor variations. At the same time, the estimated mean beat frequency is passed to an additional signal processing block, together with the frequency output of the first RST, in order to provide an improved estimate of the prime component frequency. This can be achieved by calculating the moving average of the prime component frequency, where the moving average is calculated over the period of the mean beat frequency, in order to minimise the variation induced by the beat frequency.

Figure 8 shows the simulated output resulting from the signal processing scheme shown in Figure 7. The upper plot shows the calculated mean value of the raw beat frequency. Prior to the fault at t = 100 s, the mean frequency shows random variation. After the onset of the fault, the mean beat frequency rapidly settles on 28 Hz. The middle plot shows the calculated standard deviation of the raw (n.b. not the mean) beat frequency. Prior to the onset of the fault, the standard deviation is typically around 10 Hz. After the fault, the standard deviation drops steadily, settling at around 0.1 Hz. A threshold level for detecting the onset of an interference component has been set at 0.8 Hz in this case. The lower plot shows the flag output of the diagnostic test, with two possible states: either the original signal is clean, or an Interference component (i.e. a fault) has been detected. As long as the standard deviation of the beat

frequency remains above the selected limit of 0.8 Hz, the diagnostic state value is set to ‘Clean’. At approximately t = 100.4 s, when the standard deviation of the beat frequency drops below the threshold, the flag is set to ‘Interference’.

Once the presence of an interfering component has been detected, additional Prism-based signal processing blocks may be created to estimate the parameters of the fault and to provide a correction to the measurement. Figure 9 shows an extension of Figure 7 in which a Dynamic Notch Filtering block (as explained in [4]) is introduced to isolate and track both the Primary component and the Interference component. The Dynamic Notch Filtering block requires the input signal and the estimated frequencies of the two components.

Figure 10 shows the calculated frequency estimate of the Primary component generated by the signal processing scheme of Figure 9. The upper plot shows the time sequence: a step change in behaviour occurs at t = 100 s, when the Interference component is manifest, and at approximately t = 100.4 s when the correction is enabled. The lower plot shows the magnitude of the frequency error on a logarithmic scale. Prior to the onset of the Interference component the frequency error is approximately within the range ± 1e-4 Hz; after the onset of the Interference component the frequency error range increases to approximately ± 5e-2 Hz; after the correction for interference is applied the frequency error range reduces to approximately ± 5e-4 Hz, demonstrating the effectiveness of the dynamic notch filtering correction.

Similarly effective corrections are provided for frequency and phase. For example, Figure 11 shows the amplitude estimate of the Primary component. The upper plot shows the time sequence: changes in behaviour occur at t = 100 s, when the Interference component is manifest, and at approximately t = 100.4 s when the correction is enabled. The lower plot shows the magnitude of the amplitude error on a logarithmic scale. Prior to the onset of the fault the amplitude error is approximately ± 5e-6 V; after the onset of the fault the error

increases to ± 1e-3 V; after the correction strategy is applied the error reduces to approximately ± 1e-4 V.

Figure 12 shows the estimate value of the frequency of the Interference component. In the upper plot the time sequences of the estimated and true values of the frequency are given. Both are zero before the onset of the interference component at t = 100s. Approximately 0.4 s elapses before the interference component is detected, the new signal processing blocks are instantiated, and an estimate of the frequency of the interference component is provided. The lower plot shows the error in the frequency estimate, which after t = 100.4 s is within the approximate range ± 1e-3 Hz. Similar accurate tracking results are obtained for the amplitude and phase of the Interference component arising from the dynamic notch filtering and tracking calculation.

Once the signal processing scheme of Figure 9 is instantiated, it may continue to operate for as long as the fault is deemed to persist. For example if the amplitude of the Interference component drops below a certain threshold, or the standard deviation of the beat frequency rises above another threshold, it may be determined by the system that the Interference component need no longer be tracked and the associated signal processing may be discontinued. Additionally, if it is deemed that the current signal processing scheme or associated parameters are no longer suited to tracking the current set of signal components, then new signal processing blocks may be instantiated to better match the current signal properties. Note that such instantiation and filter warmup may occur in parallel with the continued operation of the current signal processing arrangement in order to prevent any interruption in tracking function before switching to the new signal processing scheme.

3. Conclusions

The Prism is a new signal processing object which has the desirable properties of a recursive calculation and simple design. It is particularly well-

suited to the new metrology challenges of the 21st Century, where the Internet of Things anticipates ubiquitous, adaptable and low cost sensing.

Because Prism design and instantiation is simple, a signal processing system can create additional processing blocks as required in response to newly detected conditions, as the sensor validation example illustrates. Furthermore, the ability to create Dynamic Notch Filtering blocks as a means of ameliorating the effects of unwanted components and/or tracking such components for the purpose of further diagnostic analysis, provides a powerful set of strategies that are suitable for implementation by automated systems. Additionally, because the design effort required to design new Prism-based signal processing is low (for example compared with conventional FIR filter design), it is possible to instantiate new Prism-based signal processing blocks in real time, selecting design parameters (e.g., *m*) to match the observed signal properties.

References

1. L.D. Xu, W. He, and S.L. Wang, *IEEE Transactions on Industrial Informatics.* 2014. Vol. 10, issue 4. p. 2233–2243.

   Xu L.D., He W., Shancang Li. Internet of Things in Industries: A Survey. // *IEEE Transactions on Industrial Informatics.* 2014. Vol. 10, № 4. P. 2233–2243.

2. B. Vogel-Heuser, D. Hess. *IEEE Transactions on Automation Science and Engineering*. 2016. Vol 14, issue 2.

   2. Vogel-Heuser B., Hess D. Guest Editorial Industry 4.0–Prerequisites and Visions // *IEEE Transactions on Automation Science and Engineering*. 2016. Vol 13. № 2. P. 411-413.

3. Sapozhnikova K.V., Henry M., Taimanov R.E. The need for standards for self-diagnosable and self-certifiable instrumentation, Datchiki & Systemi (Sensors & Systems), No 6, pp 51-57, 2006.

4. M. Henry, O.Bushuev, O. Ibryaeva. “Prism Signal Processing for Sensor Condition Monitoring”, *IEEE International Symposium on Industrial Electronics (ISIE 2017)*, Edinburgh, UK, June 2017.

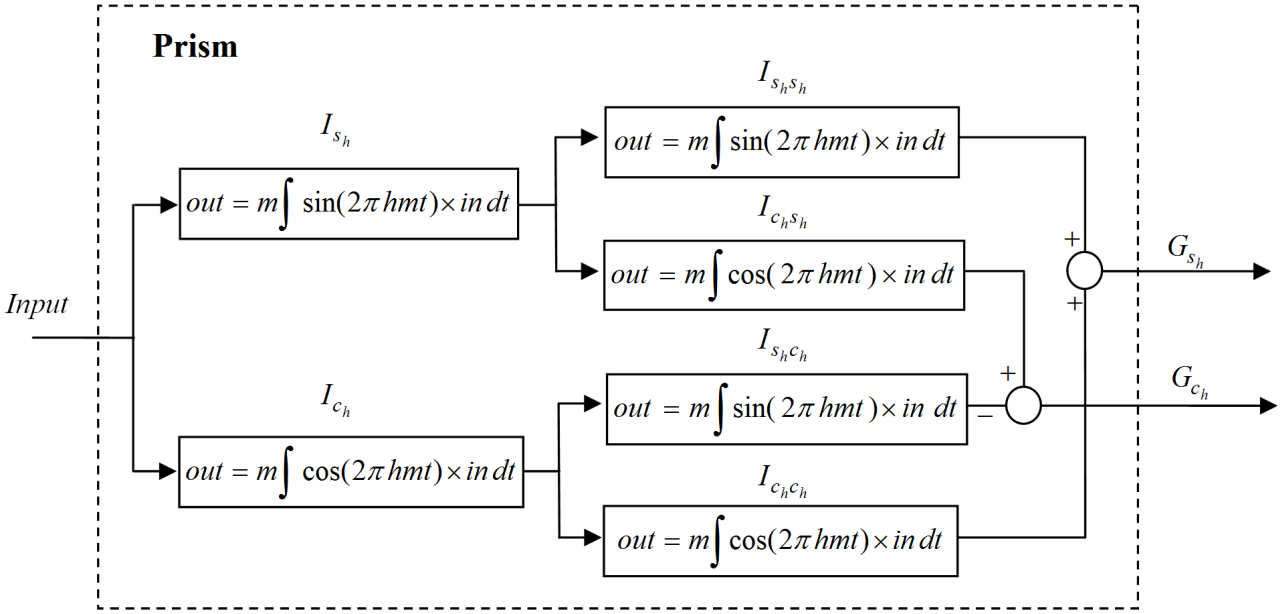

Fig. 1. Prism signal processing block with time series input and time series outputs $G_s(t)$ and $G_c(t)$. The design parameters are m the characteristic frequency, and h the harmonic number.

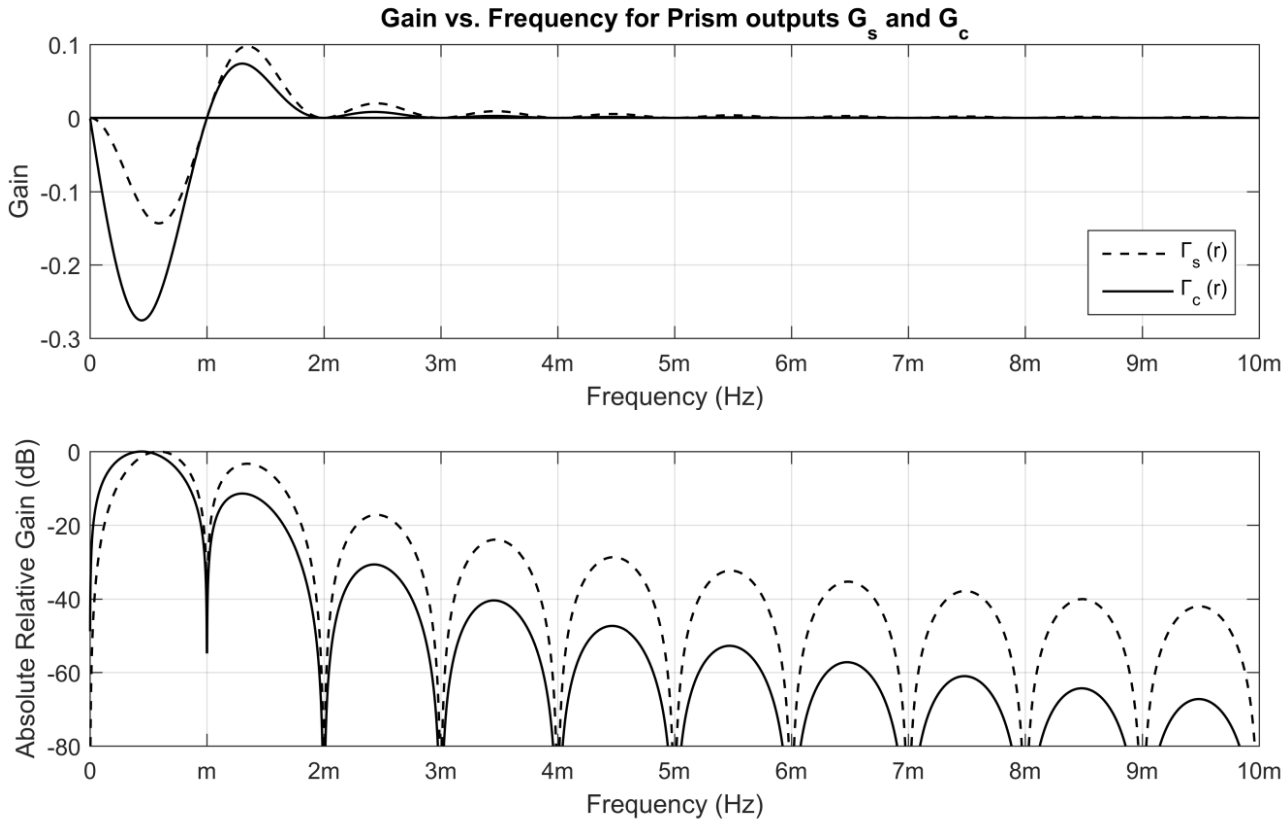

Figure 2. Gains of Prism outputs $G_s(t)$ and $G_c(t)$ with harmonic number $h = 1$.

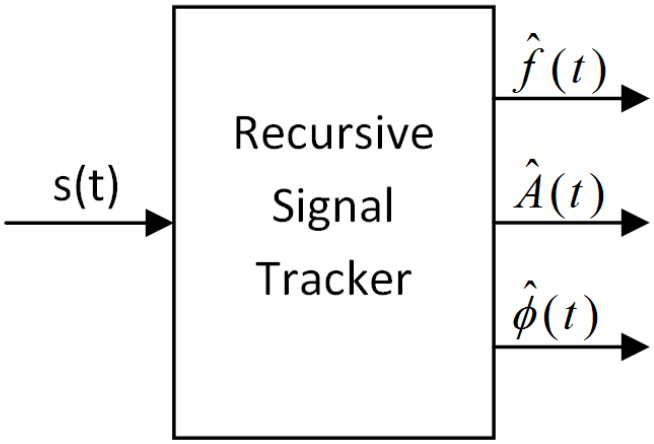

Figure 3. Sensor validation example: initial signal processing scheme. The transducer signal *s(t)* is analysed using an RST to generate sample-by-sample estimates of frequency, amplitude and phase.

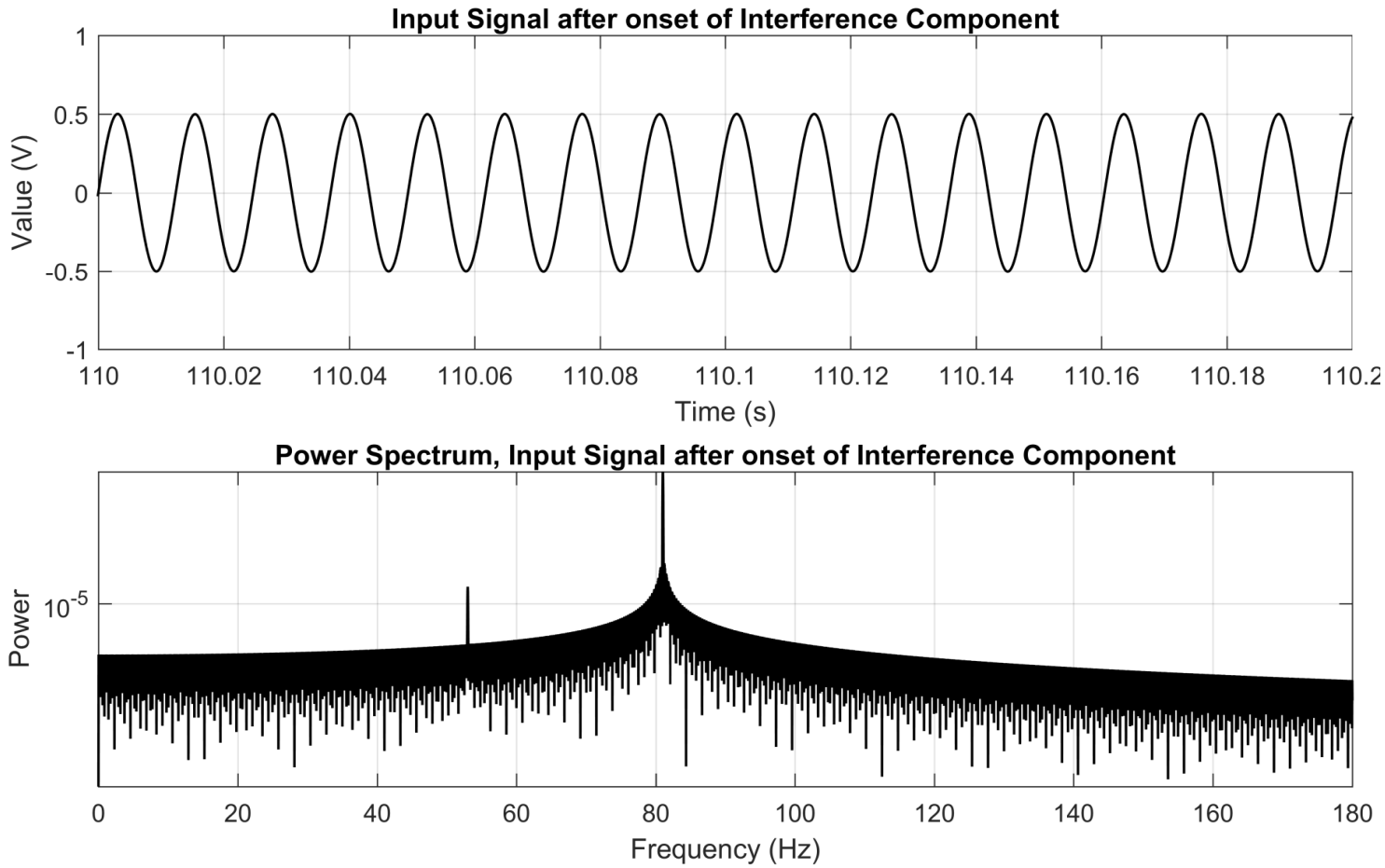

Figure 4. Raw transducer signal and power spectrum after onset of fault.

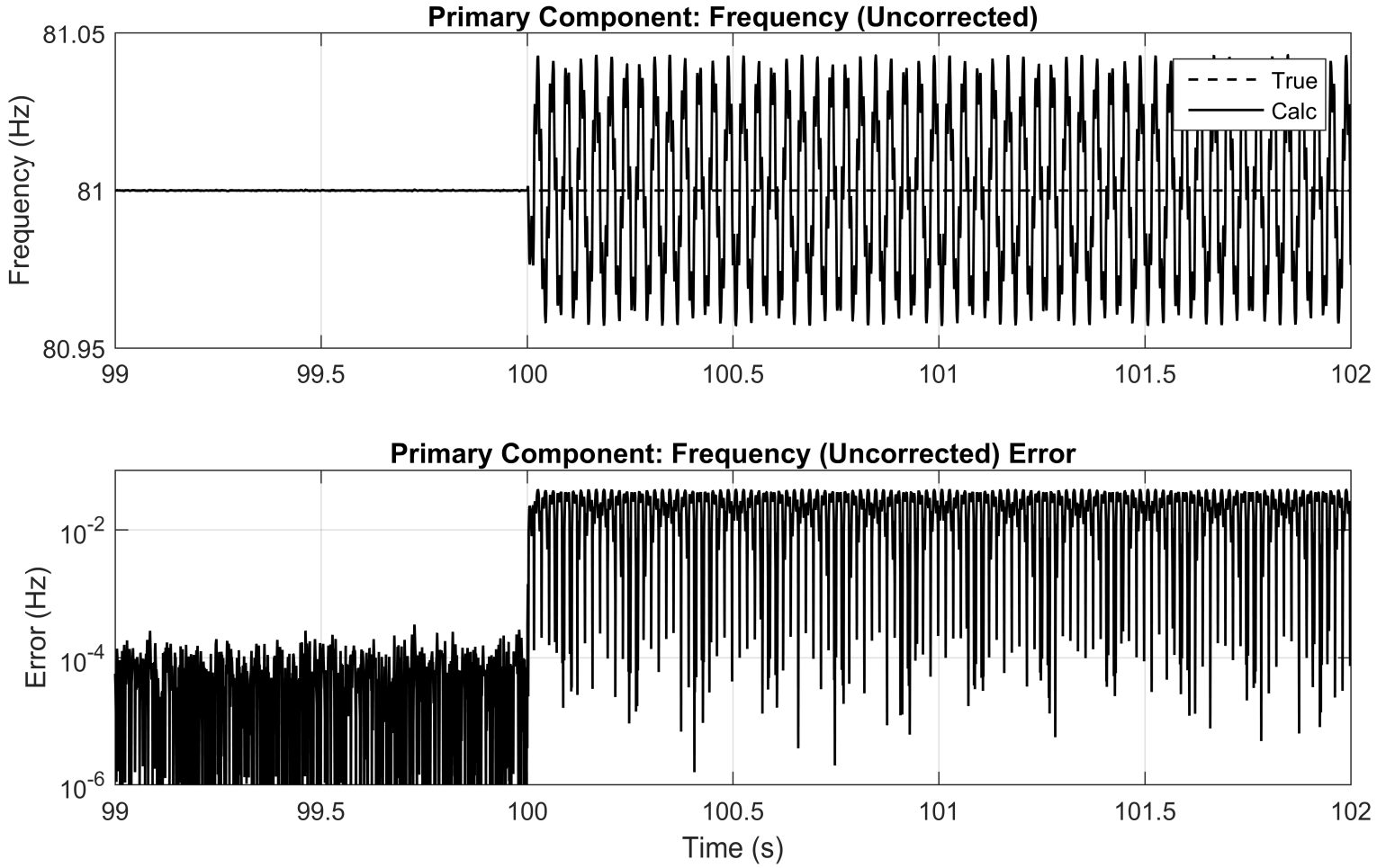

Figure 5. RST calculated value of frequency (upper) and corresponding absolute error (lower) before and after onset of fault.

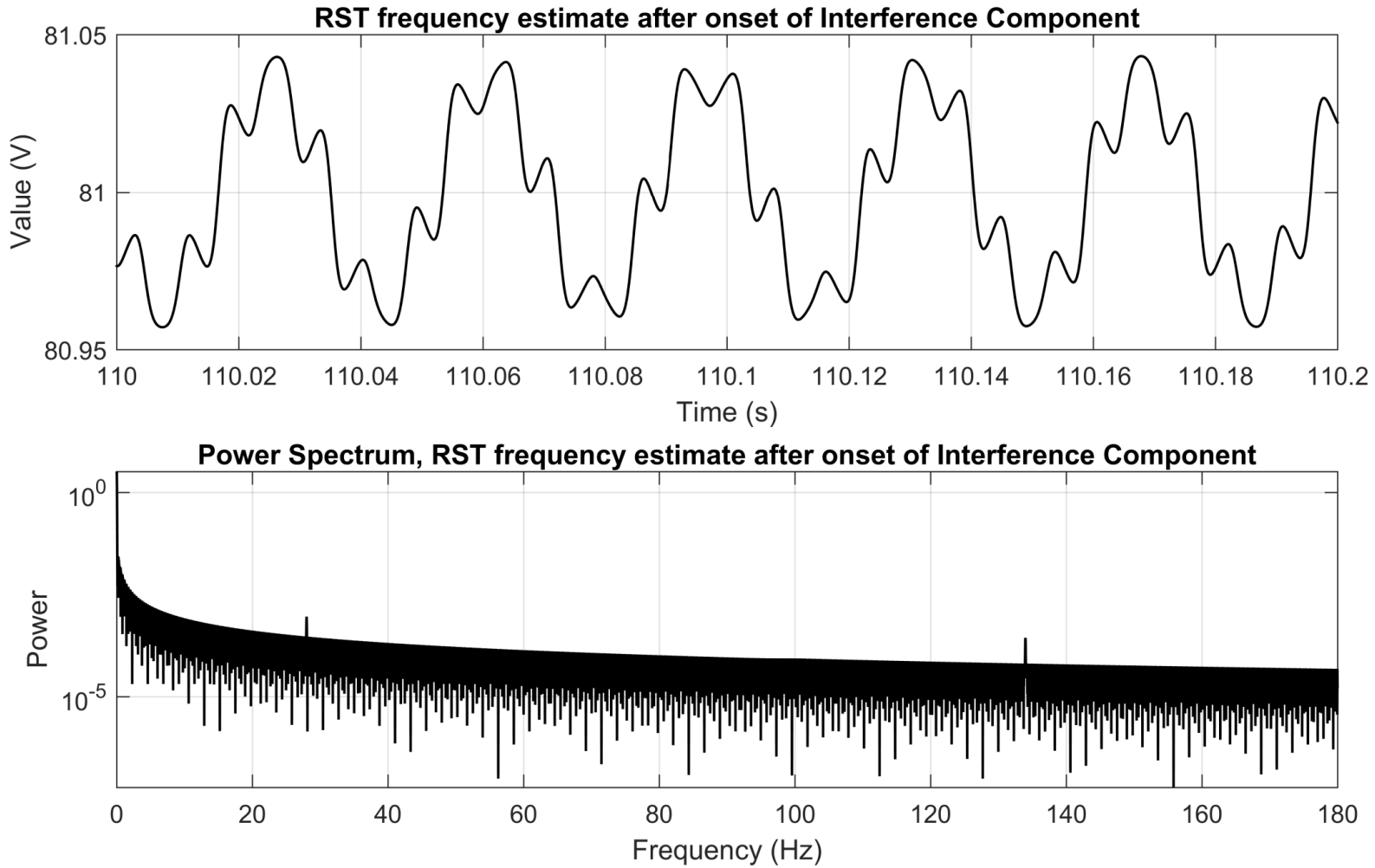

Figure 6. Detail of RST calculated value of frequency (upper); Power spectrum of frequency estimate (lower).

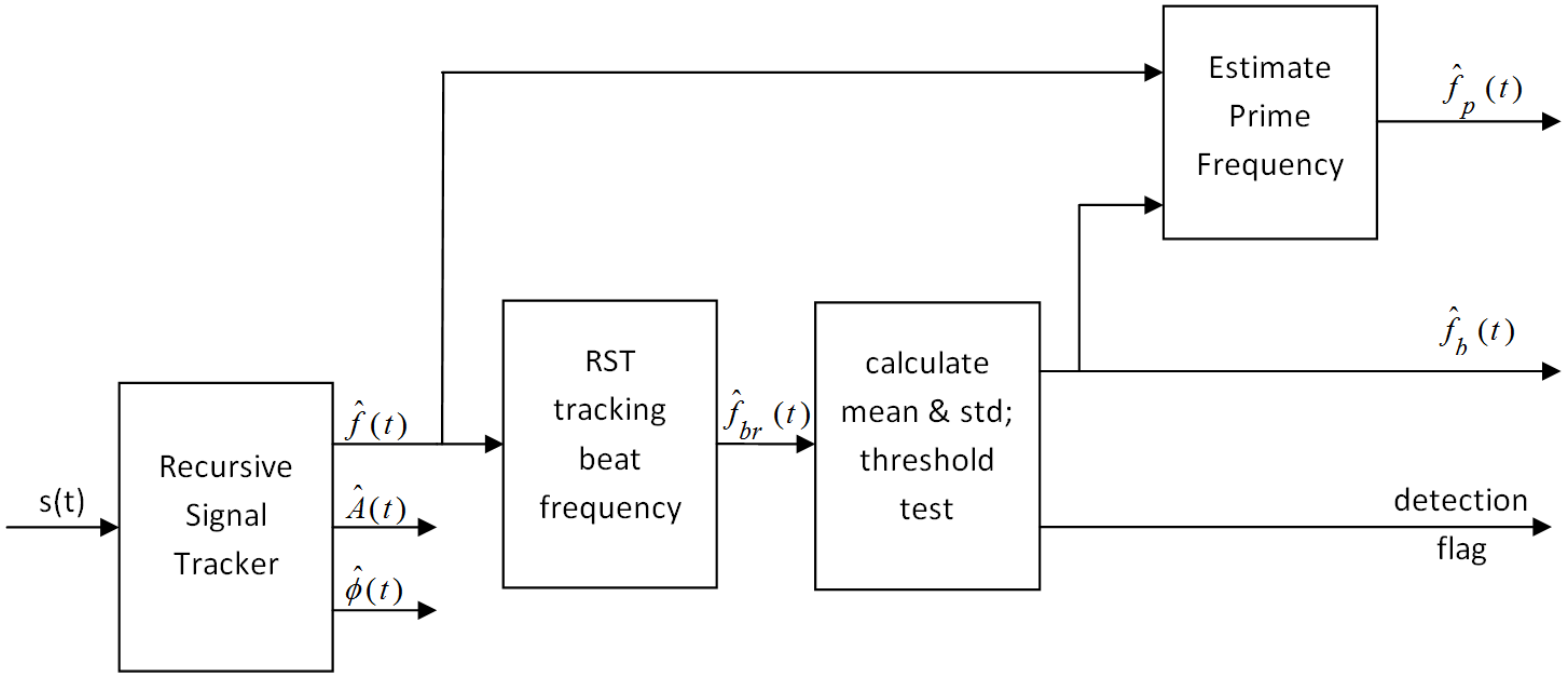

Figure 7. Signal processing scheme extended to detect interference signal.

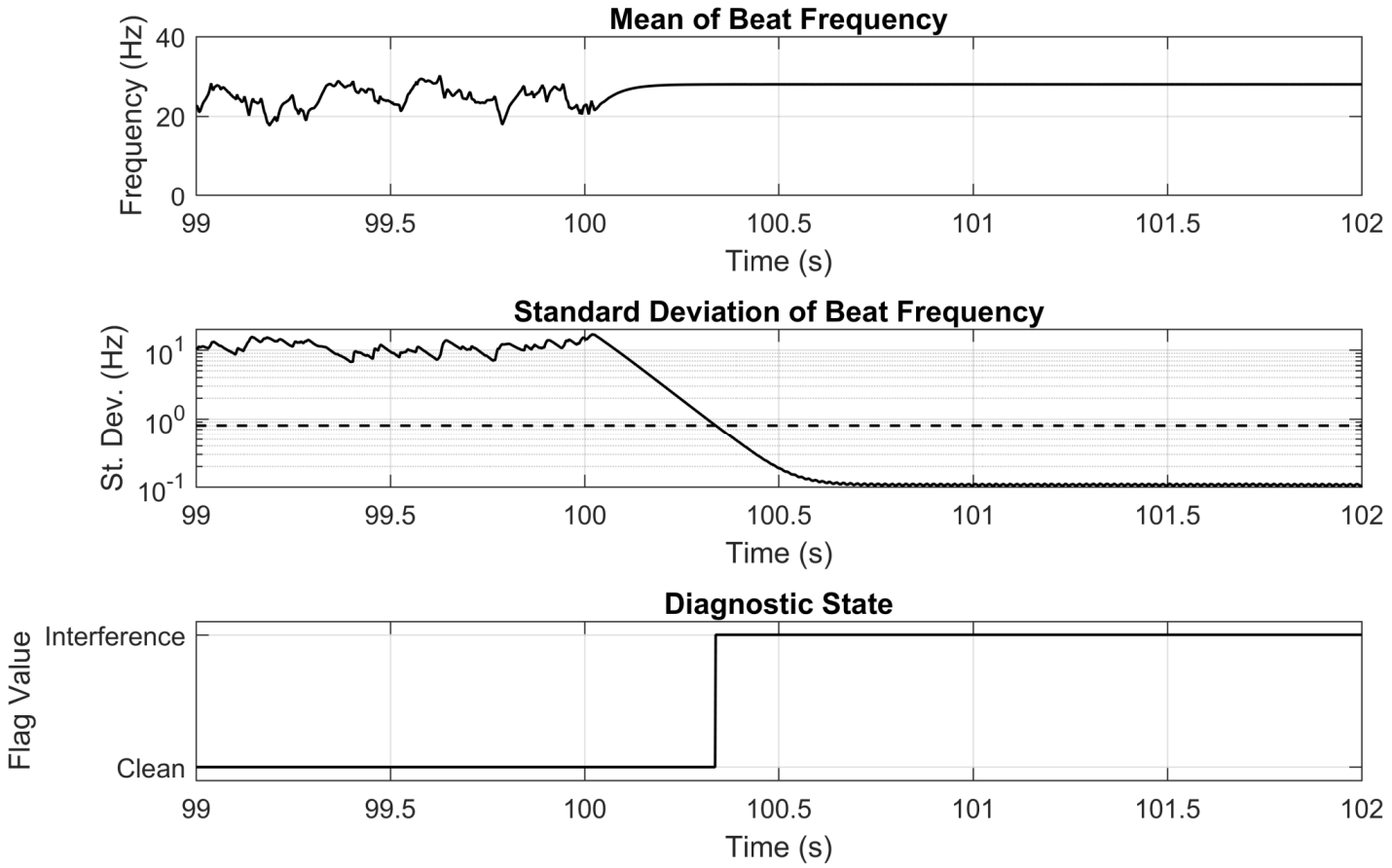

Figure 8. Simulation of onset of interference for signal processing scheme of Figure 7.

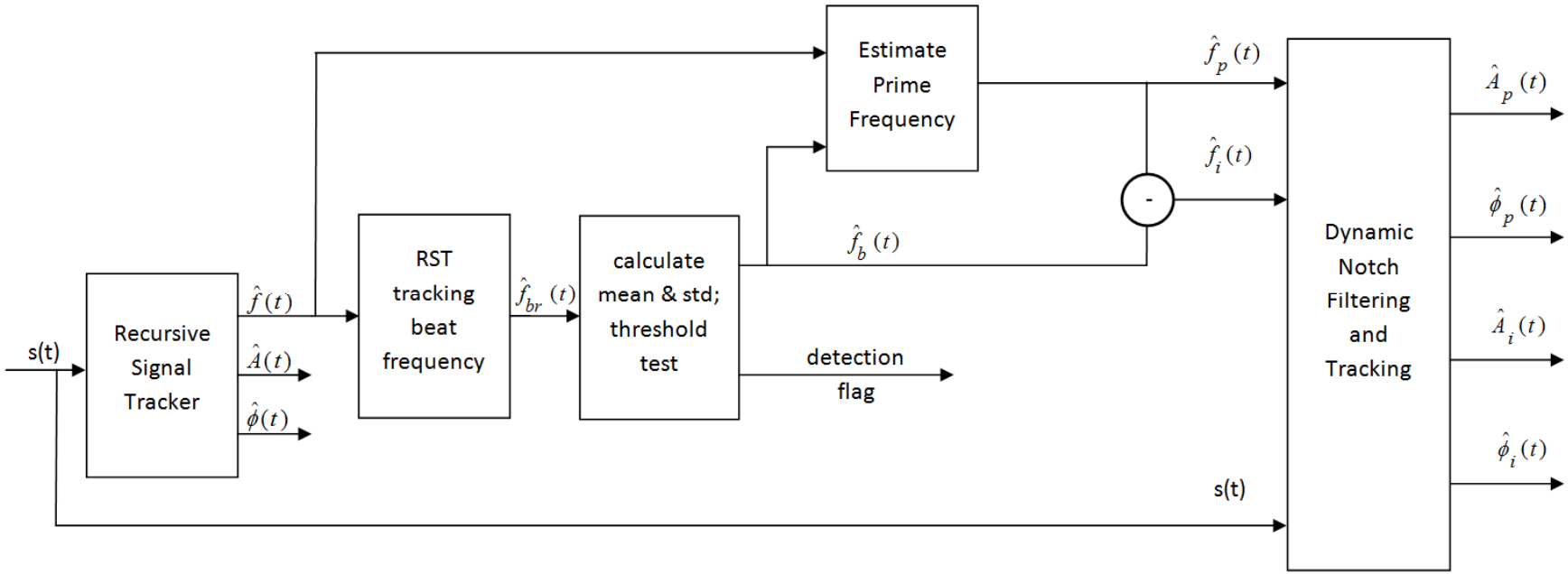

Figure 9. Signal processing scheme further extended to track and correct for interference signal.

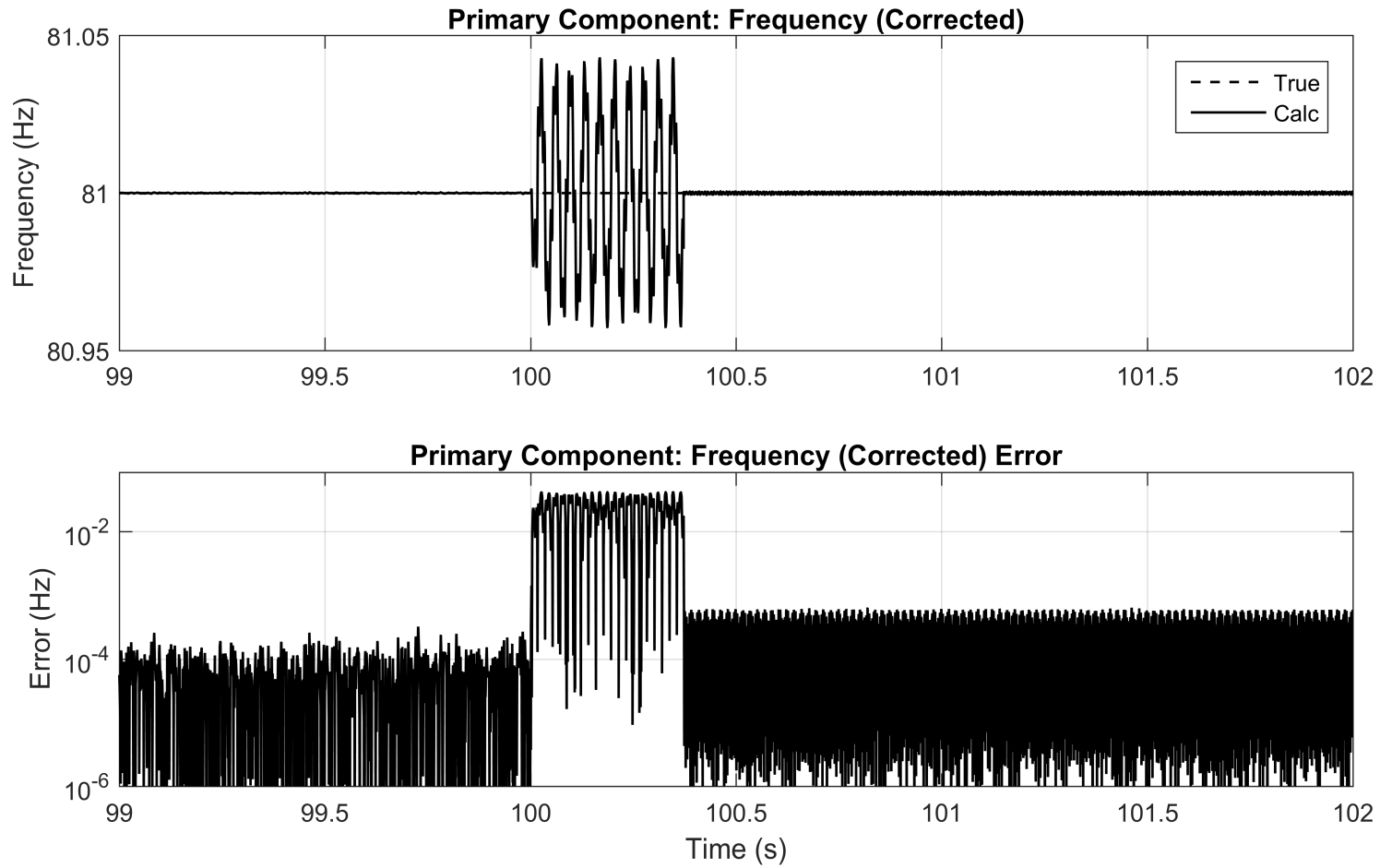

Figure 10. Corrected value of Primary Component frequency generated by signal processing scheme of Figure 9 (upper) and corresponding absolute error (lower) before and after onset of fault.

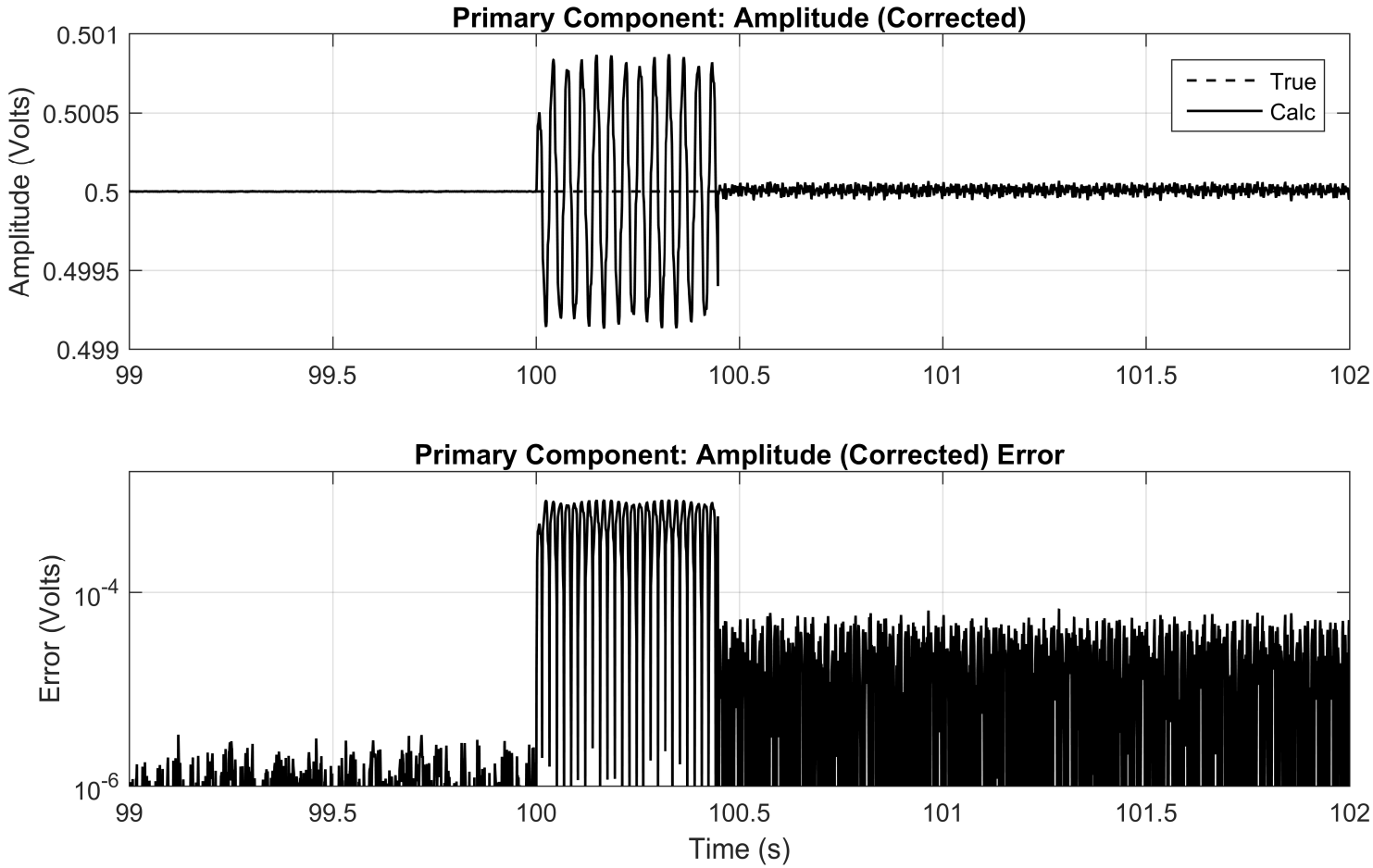

Figure 11. Corrected value of Primary Component amplitude generated by signal processing scheme of Figure 9 (upper) and corresponding absolute error (lower) before and after onset of fault.

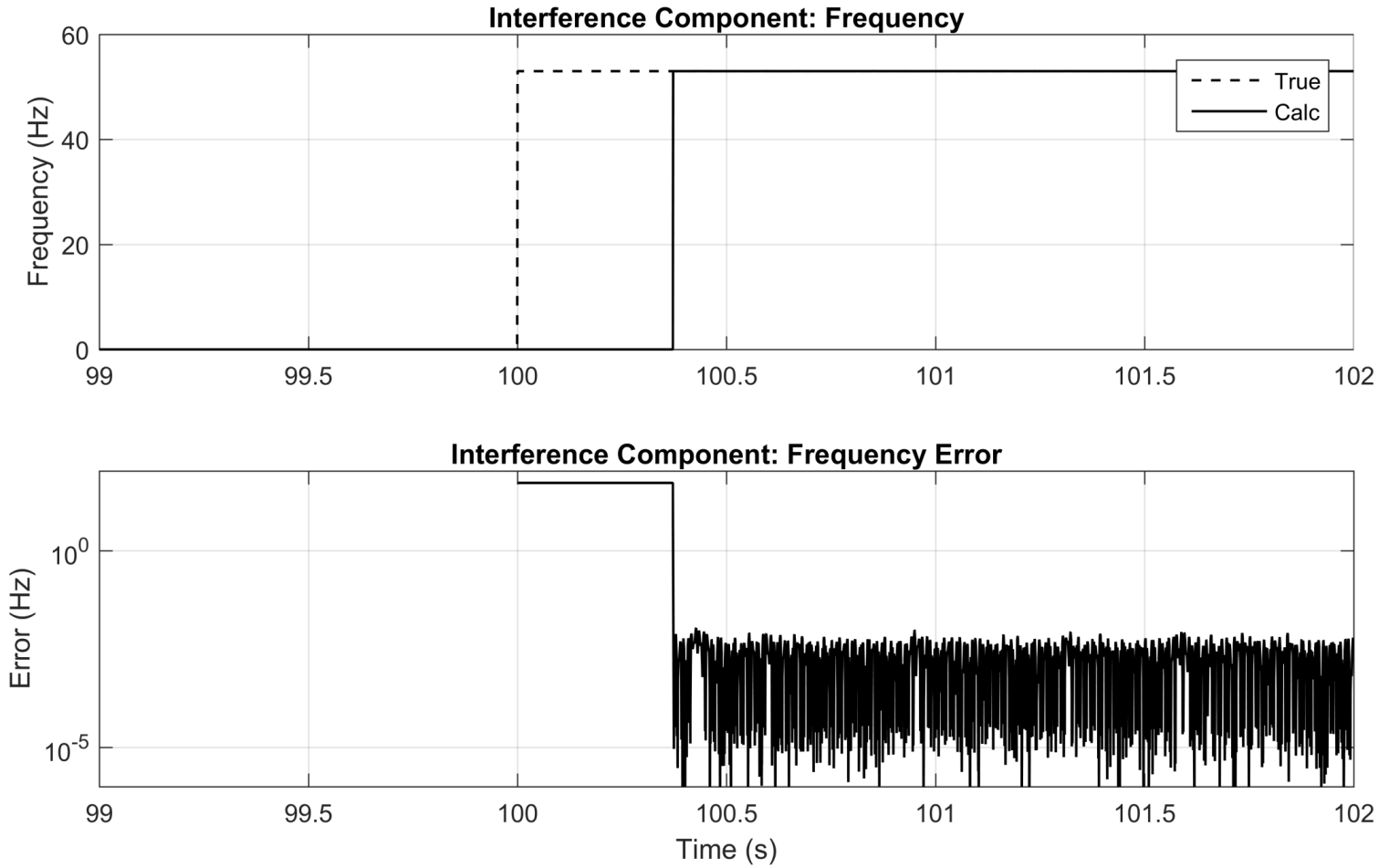

Figure 12. Estimated value of Interference Component frequency generated by signal processing scheme of Figure 9 (upper) and corresponding absolute error (lower) before and after onset of fault.